



Green University of Bangladesh

*Department of Computer Science and Engineering (CSE)
Semester: (Spring, Year: 2025), B.Sc. in CSE (Day)*

Depth-First Search (DFS) Algorithm on a Random Grid

*Course Title: Artificial Intelligence Lab
Course Code: CSE-315
Section: 221-D22*

Students Details

Name	ID
Obydullah Sayed	221902299

*Submission Date: 25-02-2025
Course Teacher's Name: Md. Sabbir Hosen Mamun*

[For teachers use only: **Don't write anything inside this box**]

<u>Lab Project Status</u>	
Marks:	Signature:
Comments:	Date:

Contents

1	Introduction	2
2	Objectives	2
3	Procedure	2
4	Code	2
5	Output	4
6	Conclusion	5
7	Github	5

1 Introduction

This lab focuses on implementing the Depth-First Search (DFS) algorithm to find a path from a source to a goal on a randomly generated grid. The grid consists of obstacle (0) and open (1) cells. The algorithm explores the grid step by step, avoiding blocked cells and revisiting the same cell. The goal is to check if a valid path exists and display the path and the order in which cells are visited.

2 Objectives

- To implement the DFS algorithm for pathfinding on a randomly generated grid.
- To find and display a valid path from the source to the goal, if it exists.
- To record and display the order in which cells are visited during the search (topological order).

3 Procedure

- A random grid of size $N \times N$ is created, with cells marked as 0 (blocked) or 1 (open).
- A source and goal are randomly selected from the open cells.
- DFS starts from the source and explores neighboring cells (up, down, left, right) using a stack.
- If the goal is found, the path is traced back from the goal to the source using parent pointers.
- The grid, path (if found), and the order of visited cells are displayed.

4 Code

```
1 import numpy as np
2 import random
3
4 class Node:
5     def __init__(self, x, y):
6         self.x = x
7         self.y = y
8
9 class DFS:
10     def __init__(self):
11         self.directions = [(1, 0, "down"), (-1, 0, "up"), (0, 1, "
right"), (0, -1, "left")]
12         self.found = False
13         self.N = 0
```

```

14     self.source = None
15     self.goal = None
16     self.visited = set()
17     self.path = []
18     self.topological_order = []
19     self.parent = {}
20
21     def run(self):
22         self.N = np.random.randint(4, 8)
23         graph = np.random.randint(0, 2, size=(self.N, self.N))
24         print("Grid:")
25         print(graph)
26
27         valid_cells = []
28         for x in range(self.N):
29             for y in range(self.N):
30                 if graph[x][y] == 1:
31                     valid_cells.append((x, y))
32
33         source_x, source_y = random.choice(valid_cells)
34         dest_x, dest_y = random.choice(valid_cells)
35         while source_x == dest_x and source_y == dest_y:
36             dest_x, dest_y = random.choice(valid_cells)
37
38         self.source = Node(source_x, source_y)
39         self.goal = Node(dest_x, dest_y)
40
41         print("Source:", self.source.x, self.source.y)
42         print("Goal:", self.goal.x, self.goal.y)
43
44         self.st_dfs(graph)
45
46         if self.found:
47             print("Goal found & DFS Path:")
48             self.print_path()
49         else:
50             print("Goal cannot be reached")
51
52         print("Topological Order:", self.topological_order)
53
54     def st_dfs(self, graph):
55         stack = []
56         stack.append(self.source)
57         self.visited.add((self.source.x, self.source.y))
58         self.parent[(self.source.x, self.source.y)] = None
59
60         while stack:
61             u = stack.pop()
62             self.topological_order.append((u.x, u.y))
63
64             if u.x == self.goal.x and u.y == self.goal.y:
65                 self.found = True
66                 self.explore_path(u)
67                 return
68
69             for dx, dy, move in self.directions:
70                 v_x, v_y = u.x + dx, u.y + dy

```

```

72         if 0 <= v_x < self.N and 0 <= v_y < self.N and graph[
v_x][v_y] == 1 and (v_x, v_y) not in self.visited:
73             self.visited.add((v_x, v_y))
74             stack.append(Node(v_x, v_y))
75             self.parent[(v_x, v_y)] = (u.x, u.y, move)
76
77     def explore_path(self, node):
78         current = (node.x, node.y)
79         while current is not None:
80             parent_info = self.parent[current]
81             if parent_info is not None:
82                 parent_x, parent_y, move = parent_info
83                 self.path.append((current[0], current[1], move))
84                 current = (parent_x, parent_y)
85             else:
86                 self.path.append((current[0], current[1], None))
87                 current = None
88         self.path.reverse()
89
90     def print_path(self):
91         for x, y, move in self.path:
92             if move is not None:
93                 print(f"({x}, {y}), {move}")
94             else:
95                 print(f"({x}, {y})")
96
97 if __name__ == "__main__":
98     dfs = DFS()
99     dfs.run()

```

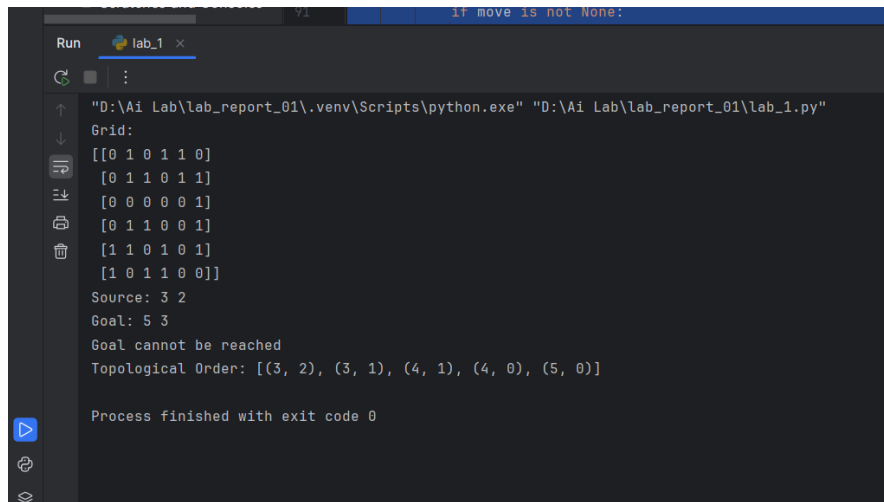
5 Output

```

Run lab_1 x
"D:\Ai Lab\lab_report_01\.venv\Scripts\python.exe" "D:\Ai Lab\lab_report_01\lab_1.py"
Grid:
[[0 0 1 0 0 0 1]
 [0 0 0 0 0 0 0]
 [0 0 1 0 1 1 0]
 [1 1 1 1 0 0 1]
 [1 1 0 1 1 1 0]
 [1 0 0 1 0 0 1]
 [1 0 0 0 1 0 0]]
Source: 5 0
Goal: 4 3
Goal found & DFS Path:
(5, 0)
(4, 0), up
(4, 1), right
(3, 1), up
(3, 2), right
(3, 3), right
(4, 3), down
Topological Order: [(5, 0), (4, 0), (4, 1), (3, 1), (3, 2), (3, 3), (4, 3)]
Process finished with exit code 0

```

Figure 1: Goal Found



```
Run lab_1 x
"D:\Ai Lab\lab_report_01\.venv\Scripts\python.exe" "D:\Ai Lab\lab_report_01\lab_1.py"
Grid:
[[0 1 0 1 1 0]
 [0 1 1 0 1 1]
 [0 0 0 0 0 1]
 [0 1 1 0 0 1]
 [1 1 0 1 0 1]
 [1 0 1 1 0 0]]
Source: 3 2
Goal: 5 3
Goal cannot be reached
Topological Order: [(3, 2), (3, 1), (4, 1), (4, 0), (5, 0)]
Process finished with exit code 0
```

Figure 2: Goal Not Found

6 Conclusion

The DFS algorithm successfully finds a path from the source to the goal on the grid if one exists. It also records the order in which cells are visited. This lab demonstrates how DFS can be used for pathfinding in grid-based problems.

7 Github

This is the GitHub repository link of the code: [https://github.com/sayed-2299/Academic/tree/main/Artificial%20Intelligence%20%20Lab \(CSE-316\) /Lab_Report_01](https://github.com/sayed-2299/Academic/tree/main/Artificial%20Intelligence%20%20Lab%20(CSE-316)/Lab_Report_01)